Attorney Docket No.: 42P8254 Patent

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re Patent Application of: Leonid Baraz, et al. Application No.: 10/043,474 Filed: 01/10/2002 For: REGISTER ALLOCATION FOR PROGRAM EXECUTION ANALYSIS	Examiner: Chow, Chih Ching Art Unit: 2191 Confirmation No.: 6444 **FIRST CLASS CERTIFICATE OF MAILING** I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail with sufficient postage in an envelope addressed to Mail Stop RCE, Commissioner for Patents, P.O. Box 1450, Alexandria, V.A. 22313-1450 on 6/5/07 Judy L. Steinkraus

Mail Stop RCE
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450

DECLARATION UNDER 37 C.F.R. § 1.131

I, Leonid Baraz, declare as follows:

1. I am one of the inventors of the above identified patent application. I am also an employee of the assignee Intel Corporation of the present application.

2. I have reviewed the application, including the claims of the application. I have also reviewed a copy of the claims set forth in Exhibit A, which are the pending claims that include amendments made by an amendment accompanying my Declaration.

3. The declaration made herein is to establish completion of the invention in the application in the United States prior to October, 2001, which is the filing date of the U.S. patent application 09/982,020 entitled "Integrated Register Allocator In A Compiler" to Markstein, et al.

4. Below stated are activities of myself and Intel Corporation regarding the date on which we conceived and reduced to practice my invention.

5. The claimed invention is embodied in Intel's products and integrated circuit devices. My fellow co-inventor and I conceived, developed, and tested a prototype of the claimed invention at least prior to October, 2001.

6. Exhibit B is a redacted Invention Disclosure document completed prior to May 2001, signed by myself and my fellow inventor, and received by the Legal Department of Intel prior to May 2001. Exhibit B demonstrates that the claimed invention was conceived, built, and tested at least prior to May 2001. The dates have been redacted on Exhibit B.

7. Exhibit C comprises a redacted photocopy of the Award Memo email with the redacted email attachment prior to October 2001 that contains the claimed invention. As evidenced by Exhibit B, the claimed invention had been conceived, built, and tested at least before October 2001. The date has been redacted on Exhibit C.

8. Exhibit D is a redacted photocopy of the Notification of Patent Application Filings email prior to October 2001, which refers to the claimed invention. As evidenced by this document, the claimed invention had been conceived, built, and tested at least before October 2001. The date has been redacted on Exhibit D.

9. Exhibit E is a redacted photocopy of the email containing discussions on the claimed invention prior to October 2001. As evidenced by this document, the claimed invention had been conceived, built, and tested at least before October 2001. The date has been redacted on Exhibit E.

10. On January 10, 2002, the instant patent application S/No.10/043,474 was filed with the United States Patent and Trademark Office, thereby constructively reducing the claimed invention to practice.

I declare, to the best of my knowledge, that all statements made in this document are true, and that all statements made on the information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under § 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the above-identified patent application or any patent issued thereon.

Dated: 11-MAY-2007

Name: Leonid Baraz

EXHIBIT A

CLAIMS

1. A machine-implemented method comprising:

analyzing one or more instructions of a first program; and

modifying the first program to expand a register set for a routine in the first program transparently to execution of the first program that includes adding one or more registers to the register set, wherein the one or more registers of an expanded register set are used by a second program to store data used to analyze the execution of the first program.

2. The method of claim 1, comprising:

identifying one or more register moves for the expanded register set; and

modifying the first program to perform the identified one or more register moves.

3. The method of claim 2, wherein the identifying comprises:

defining one or more move chains for the expanded register set, and

identifying a sequence of one or more register moves based on the defined one or more move chains.

4. The method of claim 1, wherein the modifying the first program comprises modifying the first program to expand a register set for a callee routine of the first program.

5. The method of claim 4, comprising:

modifying the first program to expand a register set for a caller routine that is to call the callee routine.

6. The method of claim 5, wherein the modifying the first program to expand a register set for the callee routine comprises modifying the first program to expand a register set that includes one or more registers of the register set for the caller routine.

7. The method of claim 5, comprising:

identifying one or more register moves for the register set of the caller routine; and

modifying the first program to perform the identified one or more register moves prior to or upon returning from the callee routine to the caller routine.

8. The method of claim 5, comprising:

identifying a register move from a register added to the register set for the caller routine to a register added to the register set for the callee routine; and

modifying the first program to perform the identified register move.

10. A machine-readable medium having instructions that, if executed by a machine, cause the machine to perform a method comprising:

analyzing one or more instructions of a first program; and

modifying the first program to expand a register set for a routine in the first program transparently to execution of the first program that includes adding one or more registers to the register set, wherein the one or more registers of an expanded register set are used by a second program to store data used to analyze the execution of the first program.

11. The machine-readable medium of claim 10, wherein the method comprises:

identifying one or more register moves for the expanded register set; and

modifying the first program to perform the identified one or more register moves.

12. The machine-readable medium of claim 11, wherein the identifying comprises:

defining one or more move chains for the expanded register set, and

identifying a sequence of one or more register moves based on the defined one or more move chains.

13. The machine-readable medium of claim 10, wherein the modifying the first program comprises modifying the first program to expand a register set for a callee routine of the first program.

14. The machine-readable medium of claim 13, wherein the method comprises:

modifying the first program to expand a register set for a caller routine that is to call the callee routine.

15. The machine-readable medium of claim 14, wherein the modifying the first program to expand a register set for the callee routine comprises modifying the first program to expand a register set that includes one or more registers of the register set for the caller routine.

16. The machine-readable medium of claim 14, wherein the method comprises:

identifying one or more register moves for the register set of the caller routine; and

modifying the first program to perform the identified one or more register moves prior to or upon returning from the callee routine to the caller routine.

17. The machine-readable medium of claim 14, wherein the method comprises:

identifying a register move from a register added to the register set for the caller routine to a register added to the register set for the callee routine; and

modifying the first program to perform the identified register move.

19. A system comprising:

a processor to execute instructions; and

a medium having instructions to analyze one or more instructions of a first program and to modify the first program to expand a register set for a routine in the first program transparently to execution of the first program that includes adding one or more registers to the register set, wherein the one or more registers of an expanded register set are used by a second program to store data used to analyze the execution of the one or more instructions of the first program.

20. The system of claim 19, the medium having instructions to identify one or more register moves for the expanded register set and to modify the first program to perform the identified one or more register moves.

21. The system of claim 20, the medium having instructions to define one or more move chains for the expanded register set and to identify a sequence of one or more register moves based on the defined one or more move chains.

22. The system of claim 19, the medium having instructions to modify the first program to expand a register set for a callee routine of the first program.

23. The system of claim 22, the medium having instructions to modify the first program to expand a register set for a caller routine that is to call the callee routine.

24. The system of claim 23, the medium having instructions to modify the first program to expand a register set that includes one or more registers of the register set for the caller routine.

25. The system of claim 23, the medium having instructions to identify one or more register moves for the register set of the caller routine and to modify the first program to perform the identified one or more register moves prior to or upon returning from the callee routine to the caller routine.

26. The system of claim 23, the medium having instructions to identify a register move from a register added to the register set for the caller routine to a register added to the register set for the callee routine and to modify the first program to perform the identified register move.

EXHIBIT B

12620

REDACTED
INTEL INVENTION DISCLOSURE

INTEL CONFIDENTIAL

[redacted] P8254

DATE: [redacted] Architecture/ MPG

It is important to provide accurate and detailed information on this form. The information will be used to evaluate your invention for possible filing as a patent application. When completed, please return this form to the **Legal Department at JF3-147**. If you have any questions, please call **264-0444 or 264-1476.**

1. Inventor: Baraz (Last Name) Leonid (First Name) (Middle Initial)

Phone ____+972-4-8655546_____ M/S: ___IDC-1D___ Fax # +972-4-8655938_____

Citizenship: _____ISRAEL_____ WWID ____10122409_____

Home Address: HaOren 81, Givat-Ram, _____ City _____ Kiryat-Ata_____ State _ISRAEL__ Zip _____

Group: (e.g. TMG, NBG, CEG) ____Btrans_____Division Name ____MPG_____ Subdivision _____

Supervisor*___Yaron Sheffer_____ WWID _____10022347_____ Phone +972-4-8655759 M/S: __IDC-1D__

Inventor: Devor (Last Name) Tevi (First Name) (Middle Initial)

Phone ____+972-4-8655675_____ M/S: ___IDC-1D___ Fax # +972-4-8655938_____

Citizenship: ___ISRAEL_____ WWID _____10022305_

Home Address: __Hashoftim 9a____ City ____Zichron Yaacov State ISRAEL Zip _____

Group: (e.g. TMG, NBG, CEG) _____Btrans____Division Name ____MPG__ Subdivision _____

Supervisor*___Yaron Sheffer_____ WWID _____10022347_____ Phone +972-4-8655759 M/S: __IDC-1D__

(PROVIDE SAME INFORMATION AS ABOVE FOR EACH ADDITIONAL INVENTOR)

2. Title of Invention: Method for Adding Integer Registers to an Existing Register Stack Frame in IA64

3. What technology/product/process (code name) does it relate to (be specific if you can):
Dynamic Optimization, Dynamic instrumentation

4. Stage of development (i.e. % complete, simulations done, test chips if any, etc.)
Implemented in prototype Dynamic Optimizer that is under development

5. (a) Has a description of your invention been, or will it shortly be, published outside Intel:

NO: __NO__ YES: _____ If YES, was the manuscript submitted for pre-publication approval? _____

IDENTIFY THE PUBLICATION AND THE DATE PUBLISHED: _____

(b) Has your invention been used/sold or planned to be used/sold by Intel or others?

NO: __NO__ YES: _____ DATE WAS OR WILL BE SOLD: _____

(c) Does this invention relate to technology that is or will be covered by a SIG (special interest group)/standard/ or specification?

NO: __NO__ YES: _____ Name of SIG/Standard/Specification: _____

(d) If the invention is embodied in a semiconductor device, actual or anticipated date of tapeout? _____

(e) If the invention is software, actual or anticipated date of any beta tests outside Intel None_____

6. Was the invention conceived or constructed in collaboration with anyone other than an Intel blue badge employee or in performance of a project involving entities other than Intel, e.g. government, other companies, universities or consortia? NO: __NO__ YES: _____ Name of individual or entity: _____

7. Is this invention related to any other invention disclosure that you have recently submitted? If so, please give the title and inventors: __NO_____ _____ _____ _____

[redacted] **PLEASE READ AND FOLLOW THE DIRECTIONS ON HOW TO WRITE A DESCRIPTION OF YOUR INVENTION**

Please attach a page to this form, DATED AND SIGNED BY AT LEAST ONE PERSON WHO IS NOT A NAMED INVENTOR, to provide a description of the invention, and include the following information:

1. **Describe in detail what the components of the invention are and how the invention works.**

See below

2. **Describe advantage(s) of your invention over what is done now.**

Currently there is no existing applications we know of that do this

3. **YOU MUST include at least one figure illustrating the invention. If the invention relates to software, include a flowchart or pseudo-code representation of the algorithm.**

DONE. See below

4. **Value of your invention to Intel (how will it be used?).**

Will be used in Intel IA64 dynamic optimizer.

5. **Identify the closest or most pertinent prior art that you are aware of.**

NONE

6. **Who is likely to want to use this invention or infringe the patent if one is obtained and how would infringement be detected?**

Other applications that dynamically modify executables. Would be detected by code inspection

***HAVE YOUR SUPERVISOR READ, DATE AND SIGN COMPLETED FORM**

DATE: [redacted] SUPERVISOR: Yaron Sheffer_____

BY THIS SIGNING, I (SUPERVISOR) ACKNOWLEDGE THAT I HAVE READ AND UNDERSTAND THIS DISCLOSURE, AND RECOMMEND THAT THE HONORARIUM BE PAID

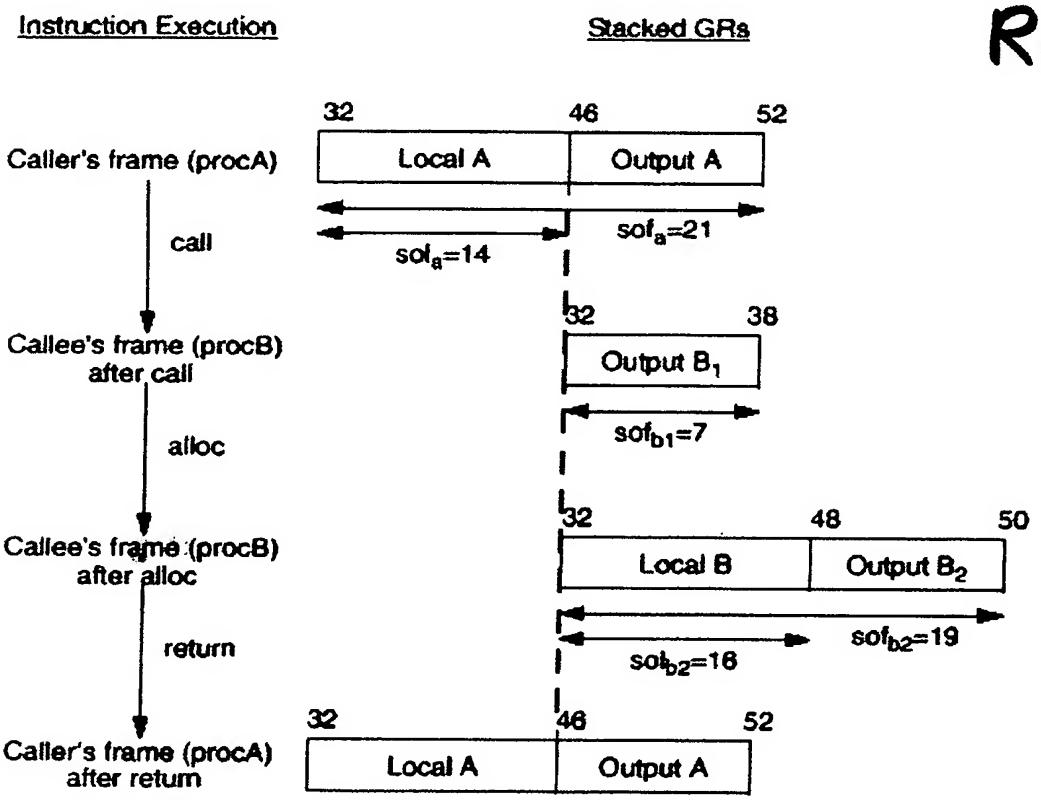
PATENT DECRIPTION: REDACTED

Applications that modify other applications (e.g. dynamic instrumenters, dynamic optimizers), may need to use registers for their own purposes in a manner that will be transparent to the source application being modified. This requires these applications to either save and restore registers in use by the source application – or to find registers that are free within a relevant scope in the source application. A method for obtaining registers in the IA64 architecture, by dynamically expanding the register frames of functions in the source application will be presented here.

The IA64 Architecture implements a per-function integer register stack. Each function may allocate itself a register stack frame of up to 96 integer stack registers beginning at GR32 through GR128. The frame is further partitioned into two variable-size areas: the local area and the output area. Immediately after a call-type branch, a new frame is created for the callee. The size of the local area of the newly activated frame is zero and the size of the output area is equal to the size of the caller's output area and overlays the caller's output area.

The callee can resize the frame using the IA64 **alloc** instruction which specifies immediate values that determine the size of the frame and the size of the locals. When a return-type branch is executed the frame reverts to the frame that was in effect at the caller before the call.

The following figure illustrates this when the function procA calls the function procB (note that *sof* is "start of frame" and *sol* is "start of locals"):



REDACTED

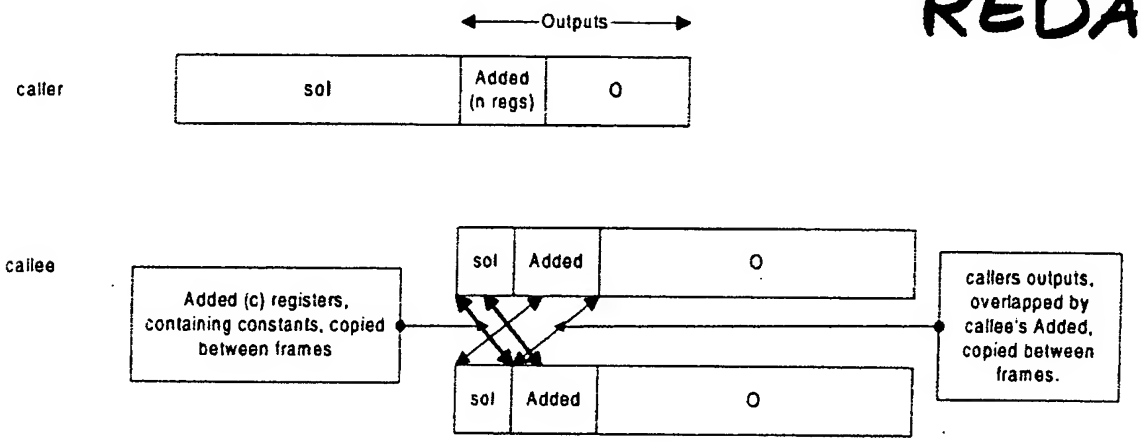
The full description of the register frame in IA64 is found at ftp://download.intel.com/design/IA64/Downloads/ADAG.pdf in section 4.1.1

As mentioned above, the modifying application will obtain registers for its own use by dynamically expanding the frame of the function in the source application. This is achieved by dynamically modifying the **alloc** instructions in the source application. The additional registers on the frame are transparent to the source application code, since these registers were not available when the source application code was generated. The output register area of the frame of each function is expanded by *n*, the number of additional registers desired. The first *n* registers in the output area of the frame are then available for use in the function. Furthermore, some of these registers may be used to hold values that are constant throughout the thread execution, and should maintain their values across calls, while others may not be required to do so. We denote by *c* the number of [illegible]ers used to hold constant values . $c < n$ and these are the first *c* registers in the added registers. This is implemented as foll

1. References to an output register must be modified to access the corresponding output register after the expansion. This is done by adding n (where n is the number of registers that were added to the output area) to each output register reference.
2. Each function must observe the output registers of its caller in the same place as they would have been before the expansion, and the caller must be presented with those registers modified by the callee in the place it expects them. E.g. if the caller's output area is 8 registers starting at r50, and 5 registers are added to the output– The implementation of paragraph (1) will cause the caller to access r55 instead of r50. The callee, however will view the callee's r50 as the first register (r32) in its frame rather than r55 as it should. Hence the callee should copy the caller's r55 into its (the callee's) r32 immediately following the **alloc**. Prior to the return, the callee should copy its r32 back into the caller's r55.
3. In order to maintain those constant values that are held in the first c registers of the additional registers, these c registers are copied from the caller frame into the proper place in the callee frame at the beginning of the callee, and from the callee frame back to the caller frame prior to return from the callee. [this was mentioned before]

Copying in the direction of the call is done immediately after the alloc in the callee. Copying in the direction of the return is done immediately before the return. This is illustrated as follows:
The diagram should mention the parameters "n" and "c".

REDACTED



As can be seen in the above diagram some registers must be copied forward in the frame and others backwards in the frame – and there can be overlap in these. An efficient copying sequence is generated by determining copy chains and then executing the copies in each chain in reverse order.
Copy chains are formed and processed as follows:

First step is to determine given sol, n and c the registers that must be copied and their detination. An array, regCopyInfoArr, whose index is the source register and contents in the destination register is used to hold this information.

```
Now build all copy chains:
regCopyInfoArrayIndex = 0
while (1)
   /* find next element in regCopyInfoArray that is not on chain */
   while (regCopyInfoArrayIndex < maximum index in regCopyInfoArr &&
          (src register ,whose number is regCopyInfoArrayIndex, is not on a copy chain ||
           this src register does not have to be copied))
      regCopyInfoArrayIndex++;
   if (regCopyInfoArrayIndex exceeds max) /* all registers on copy chains */
      break out of while(1)
   BuildOneCopyChain starting from src register ,whose number is regCopyInfoArrayIndex

BuildOneCopyChain:
set src_register to the register specified by regCopyInfoArrayIndex
while (destination_register of src_register is not 0 and is not on chain)
   add destination_register of src_register, specified by
   regCopyInfoArr[regCopyInfoArrayIndex] to chain
   src_register = destination_register
if exited loop because destination_register was already on a chain and it was on this chain
   this is a loop chain
else
   the destination_register does not have to be copied and can be used in the sequence of copies    loop
```

- copy chain

- Now each chain is used to generate a sequence of copies:
  the copies are executed in reverse order on the chain –
    the second last register is moved to the last, the third last to the second last and so forth.
    if the copy chain is a loop
      the last register of one of the non loop copy chains is used to initiate the copy sequence

The destination of the IA 64 alloc instruction may be one of the local registers that is to be overwritten by caller's output – in this case the destination of the alloc instruction will be modified to be that output register that will be written to the local specified in the alloc instruction.

REDACTED

EXHIBIT C

[redacted]

From: patent.database.group@intel.com
Sent: [redacted]
To: Devor, Tevi
Subject: AWARD MEMO

To: TEVI DEVOR
E-mail: tevi.devor@intel.com
Employee No.: 10022305

From: Leo Novakoski
Phone: (408) 765-5334

Subject: AWARD PAYMENTS

Thank you for your recent submission of the following invention disclosure:

METHOD FOR ADDING INTEGER REGISTERS TO AN EXISTING REGISTER STACK FRAME IN IA64
Disclosure #: 12620 Award, in the amount of $100*

IP Committee assigned: ARCHITECTURE

By copy of this letter, we are requesting that employee services provide you with award in the amount of $100 (tax protected)*

Your invention(s) will be reviewed at the next quarterly intellectual property committee meeting for your group to determine whether a patent application(s) will be filed. You will be notified of the outcome of this evaluation as soon thereafter as possible. If you have any questions regarding this award or Intel's patent process, please contact Janice Boulden at 503-264-0444.

*Please note that if you signed up to have all of your employee payables auto-deposited, then this award will be auto-deposited. Otherwise, this will be issued as a separate "live" check mailed to your home address.

PLEASE NOTE: Our "new" electronic inventor notifications save Intel substantial time and money but do not currently have the capability to copy your manager. Please feel free to forward to your manager.

For future update information, please visit our web site at law.intel.com/PPG2. You will need to provide your MAD log-in name and password.

nad#/username
)assword

REDACTED

[redacted]

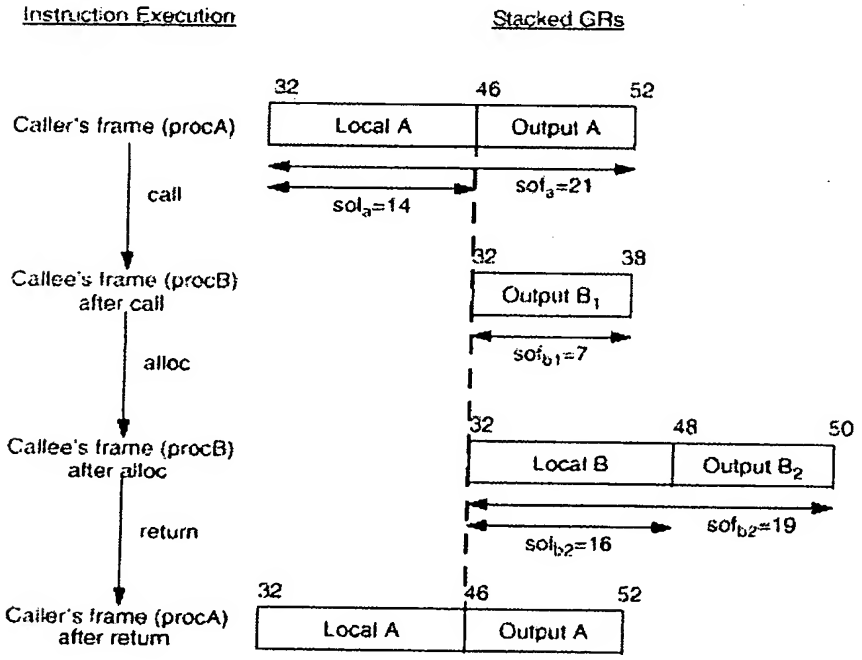
Adding Integer Registers to an Existing Register Stack Frame in IA64

Applications that modify other applications (e.g. dynamic instrumenters, dynamic optimizers), may need to use registers for their own purposes in a manner that will be transparent to the source application being modified. This requires these applications to either save and restore registers in use by the source application – or to find registers that are free within a relevant scope in the source application. A method for obtaining registers in the IA64 architecture, by dynamically expanding the register frames of functions in the source application will be presented here.

The IA64 Architecture implements a per-function integer register stack. Each function may allocate itself a register stack frame of up to 96 integer stack registers beginning at GR32 through GR128. The frame is further partitioned into two variable-size areas: the local area and the output area. Immediately after a call-type branch, a new frame is created for the callee. The size of the local area of the newly activated frame is zero and the size of the output area is equal to the size of the caller's output area and overlays the caller's output area.

The callee can resize the frame using the IA64 **alloc** instruction which specifies immediate values that determine the size of the frame and the size of the locals. When a return-type branch is executed the frame reverts to the frame that was in effect at the caller before the call.

The following figure illustrates this when the function procA calls the function procB (note that *sof* is "start of frame" and *sol* is "start of locals"):

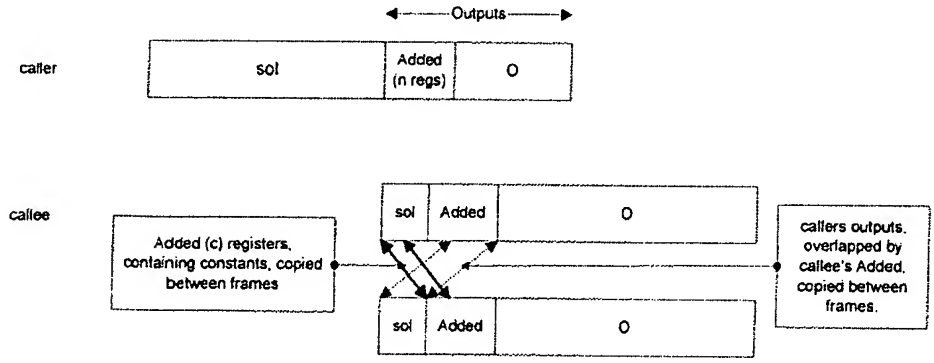


The full description of the register frame in IA64 is found at ftp://download.intel.com/design/IA64/Downloads/ADAG.pdf in section 4.1.1

As mentioned above, the modifying application will obtain registers for its own use by dynamically expanding the frame of the function in the source application. This is achieved by dynamically modifying the **alloc** instructions in the source application. The additional registers on the frame are transparent to the source application code, since these registers were not available when the source application code was generated. The output register area of the frame of each function is expanded by *n*, the number of additional registers desired. The first *n* registers in the output area of the frame are then available for use in the function. Furthermore, some of these registers may be used to hold values that are constant throughout the thread execution, and should maintain their values across calls, while others may not be required to do so. We denote by *c* the number of registers used to hold constant values . c < n and these are the first *c* registers in the added registers. This is implemented as follows:

1. References to an output register must be modified to access the corresponding output register after the expansion. This is done by adding n (where n is the number of registers that were added to the output area) to each output register reference.
2. Each function must observe the output registers of its caller in the same place as they would have been before the expansion, and the caller must be presented with those registers modified by the callee in the place it expects them. E.g. if the caller's output area is 8 registers starting at r50, and 5 registers are added to the output– The implementation of paragraph (1) will cause the caller to access r55 instead of r50. The callee, however will view the callee's r50 as the first register (r32) in its frame rather than r55 as it should. Hence the callee should copy the caller's r55 into its (the callee's) r32 immediately following the **alloc**. Prior to the return, the callee should copy its r32 back into the caller's r55.
3. In order to maintain those constant values that are held in the first c registers of the additional registers, these c registers are copied from the caller frame into the proper place in the callee frame at the beginning of the callee, and from the callee frame back to the caller frame prior to return from the callee. [this was mentioned before]

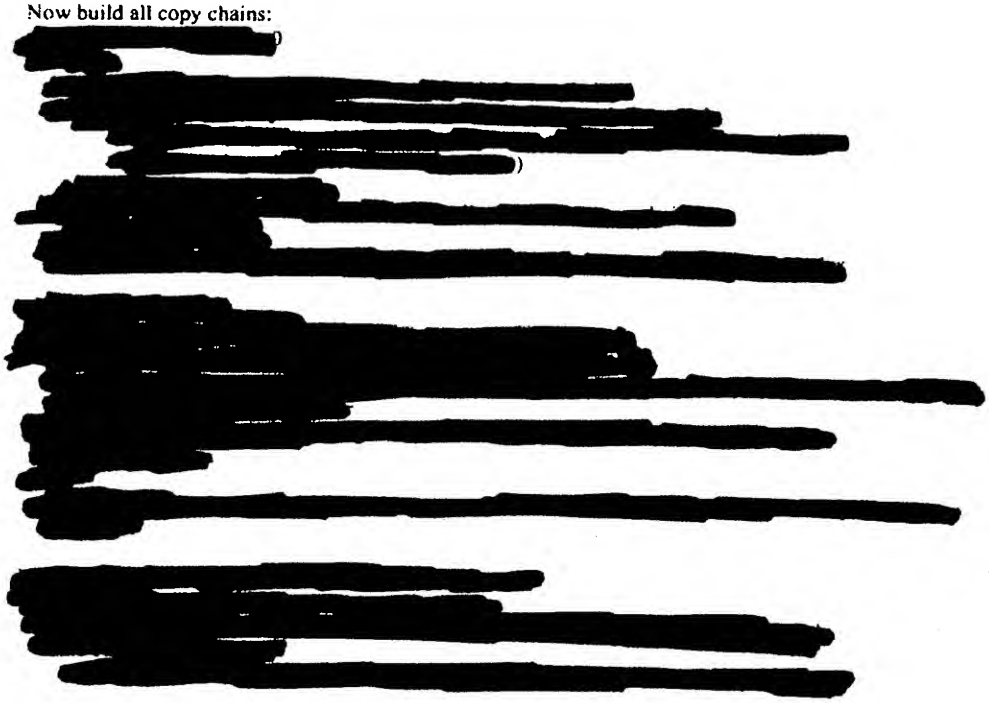
Copying in the direction of the call is done immediately after the alloc in the callee. Copying in the direction of the return is done immediately before the return. This is illustrated as follows:
The diagram should mention the parameters "n" and "c".



As can be seen in the above diagram some registers must be copied forward in the frame and others backwards in the frame – and there can be overlap in these. An efficient copying sequence is generated by determining copy chains and then executing the copies in each chain in reverse order.
Copy chains are formed and processed as follows:

First step is to determine given sol, n and c the registers that must be copied and their detination. An array, [redacted] whose index is the source register and contents in the destination register is used to hold this information.

Now build all copy chains:



The destination of the [redacted]on may be one of the local registers that is to be overwritten by caller's output – in this case the destination of the alloc instruction will be modified to be that output register that will be written to the local specified in the alloc instruction.

REDACTED

EXHIBIT D

[redacted]

From: patent.database.group@intel.com
Sent: [redacted]
To: Devor, Tevi
Subject: NOTIFICATION OF PATENT APPLICATION FILINGS

To: TEVI DEVOR
E-mail: tevi.devor@intel.com
Employee No.: 10022305

From: Leo Novakoski
Phone: (408) 765-5334

Subject: NOTIFICATION OF PATENT APPLICATION FILINGS

I am pleased to inform you that a determination has been made to file a U.S. patent application(s) covering your invention(s) as follows:

12620
METHOD FOR ADDING INTEGER REGISTERS TO AN EXISTING REGISTER STACK FRAME IN IA64

A patent attorney will be assigned to prepare the application(s) and will be contacting you for more details on your disclosure(s). Please cooperate with the attorney in answering questions and providing support for your invention(s). The attorney will use this information to prepare a draft patent application(s).

Once a draft of the application is prepared, you will be asked to review the draft to ensure that the most current version of the invention(s) is disclosed and suggest revisions prior to filing the application(s) with the U.S. Patent and Trademark Office. It is essential that you make your review of the application(s) a priority as patent rights can be lost for failure to timely file. Please do not take more than three weeks to review your application(s).

An honorarium will be paid to you once the patent application(s) is filed. In the meantime, if you have any questions, please call me.

PLEASE NOTE: Our "new" electronic inventor notifications save Intel substantial time and money but do not currently have the capability to copy your manager. Please feel free to forward to your manager.

For future update information, please visit our web site at law.intel.com/PPG2. You will need to provide your MAD log-in name and password.

nad#/username
password

REDACTED

[redacted]

EXHIBIT E

[redacted]

From: Matthew Fagan [Matthew_Fagan@bstz.com]
Sent: [redacted]
To: Devor, Tevi
Subject: RE: Patent Application

REDACTED

One hour earlier would be fine. I'll look forward to your call.

"Devor, Tevi" <tevi.devor@intel.com> [redacted]

To: "Devor, Tevi" <tevi.devor@intel.com>, Matthew Fagan/Bstz
cc: "Baraz, Leonid" <leonid.baraz@intel.com>
Subject: RE: Patent Application

Hi.

am really sorry - but I have to leave work on Monay at 17:00 our time (I
ast found out). Is it possible to do this one hour earlier on Monday, else
n Thursday?

hanks,
evi.

---Original Message-----
rom: Devor, Tevi
ent: [redacted]
o: 'Matthew Fagan'
ubject: RE: Patent Application

onid and I will call you.

vi

-Original Message-----
m: Matthew Fagan [mailto:Matthew_Fagan@bstz.com]
t: [redacted]
Devor, Tevi
leonid.baraz@intel.com
ject: RE: Patent Application

[redacted]

Hi Tevi,

That time is fine. Should I call you? Is your phone number still 972-4-8655675? If you'd prefer to call me, my direct number is 1-512-306-7655. Let me know what's easiest for you.

Matt

REDACTED

"Devor, Tevi" <tevi.devor@intel.com>[redacted]

To: Matthew Fagan/Bstz
cc: "Baraz, Leonid" <leonid.baraz@intel.com>
Subject: RE: Patent Application

Hi.

Sorry for the long delay - Is[redacted] your time ok?

Tevi.

-----Original Message-----
From: Matthew Fagan [mailto:Matthew_Fagan@bstz.com]
Sent: [redacted]
To: tevi.devor@intel.com; leonid.baraz@intel.com
Subject: Patent Application

Reminder.

---------- Forwarded by Matthew Fagan/Bstz on[redacted]
PM ----------

Matthew Fagan
[redacted]

To: tevi.devor@intel.com, leonid.baraz@intel.com
cc:
Subject: Patent Application

---------- Forwarded by Matthew Fagan/Bstz on[redacted]
M ----------

[redacted]

Matthew Fagan
[redacted]

REDACTED

To: tevi.devor@intel.com, leonid.baraz@intel.com
cc:
Subject: Patent Application

Hi Tevi and Leonid,

I am the patent attorney assigned to prepare the patent application covering your invention entitled "Method for Adding Integer Registers to an Existing Register Stack Frame in IA64". Sorry about the delay in getting back to you.

Please let me know when you would be available to talk about your invention. I believe you are eight hours ahead of us. I am in the US Central Time zone which is two hours ahead of the US Pacific time zone. I would prefer anytime from 8 to 11 AM my time which is 4 to 7 PM your time.

I look forward to hearing from you.

Matt

****
Matthew C. Fagan
BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN, L.L.P.
1501 South Mo Pac Expressway
Suite 250
Austin, Texas 78746-7596

Telephone: (512) 330-0844
Facsimile: (512) 330-0476

E-Mail: mailto:matthew_fagan@bstz.com
Website: http://www.bstz.com

****
Matthew C. Fagan
BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN, L.L.P.
1501 South Mo Pac Expressway
Suite 250
Austin, Texas 78746-7596

[redacted]

Telephone: (512) 330-0844
Facsimile: (512) 330-0476

E-Mail: mailto:matthew_fagan@bstz.com
Website: http://www.bstz.com

****

REDACTED

****

Matthew C. Fagan
BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN, L.L.P.
1501 South Mo Pac Expressway
Suite 250
Austin, Texas 78746-7596

Telephone: (512) 330-0844
Facsimile: (512) 330-0476

E-Mail: mailto:matthew_fagan@bstz.com
Website: http://www.bstz.com

****

****

Matthew C. Fagan
BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN, L.L.P.
1501 South Mo Pac Expressway
Suite 250
Austin, Texas 78746-7596

Telephone: (512) 330-0844
Facsimile: (512) 330-0476

E-Mail: mailto:matthew_fagan@bstz.com

Website: http://www.bstz.com

****

REDACTED

****

Matthew C. Fagan
BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN, L.L.P.
1501 South Mo Pac Expressway
Suite 250
Austin, Texas 78746-7596

Telephone: (512) 330-0844
Facsimile: (512) 330-0476

E-Mail: mailto:matthew_fagan@bstz.com
Website: http://www.bstz.com

****

This electronic message and its accompanying attachments (if any) contain information from the law firm of Blakely Sokoloff Taylor & Zafman LLP that is confidential and/or subject to attorney-client privilege. If you are not the intended recipient, be aware that any disclosure, copying, distribution, or use of the contents of this information is prohibited. If you have received this message in error, please notify the above attorney by telephone immediately.

[redacted]

Attorney Docket No.: 42P8254

Patent

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re Patent Application of: Leonid Baraz, et al. Application No.: 10/043,474 Filed: 01/10/2002 For: REGISTER ALLOCATION FOR PROGRAM EXECUTION ANALYSIS	Examiner: Chow, Chih Ching Art Unit: 2191 Confirmation No.: 6444 **FIRST CLASS CERTIFICATE OF MAILING** I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail with sufficient postage in an envelope addressed to Mail Stop RCE, Commissioner for Patents, P.O. Box 1450, Alexandria, VA. 22313-1450 on 6/5/07 Date Judy L. Stemkraus

Mail Stop RCE
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450

DECLARATION UNDER 37 C.F.R. § 1.131

I, Tevi Devor, declare as follows:

1. I am one of the inventors of the above identified patent application. I am also an employee of the assignee Intel Corporation of the present application.

2. I have reviewed the application, including the claims of the application. I have also reviewed a copy of the claims set forth in Exhibit A, which are the pending claims that include amendments made by an amendment accompanying my Declaration.

3. The declaration made herein is to establish completion of the invention in the application in the United States prior to October, 2001, which is the filing date of the U.S. patent application 09/982,020 entitled "Integrated Register Allocator In A Compiler" to Markstein, et al.

4. Below stated are activities of myself and Intel Corporation regarding the date on which we conceived and reduced to practice my invention.

5. The claimed invention is embodied in Intel's products and integrated circuit devices. My fellow co-inventor and I conceived, developed, and tested a prototype of the claimed invention at least prior to October, 2001.

6. Exhibit B is a redacted Invention Disclosure document completed prior to May 2001, signed by myself and my fellow inventor, and received by the Legal Department of Intel prior to May 2001. Exhibit B demonstrates that the claimed invention was conceived, built, and tested at least prior to May 2001. The dates have been redacted on Exhibit B.

7. Exhibit C comprises a redacted photocopy of the Award Memo email with the redacted email attachment prior to October 2001 that contains the claimed invention. As evidenced by Exhibit B, the claimed invention had been conceived, built, and tested at least before October 2001. The date has been redacted on Exhibit C.

8. Exhibit D is a redacted photocopy of the Notification of Patent Application Filings email prior to October 2001, which refers to the claimed invention. As evidenced by this document, the claimed invention had been conceived, built, and tested at least before October 2001. The date has been redacted on Exhibit D.

9. Exhibit E is a redacted photocopy of the email containing discussions on the claimed invention prior to October 2001. As evidenced by this document, the claimed invention had been conceived, built, and tested at least before October 2001. The date has been redacted on Exhibit E.

10. On January 10, 2002, the instant patent application S/No.10/043,474 was filed with the United States Patent and Trademark Office, thereby constructively reducing the claimed invention to practice.

I declare, to the best of my knowledge, that all statements made in this document are true, and that all statements made on the information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under § 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the above-identified patent application or any patent issued thereon.

Dated: April 13, 2007 Tevi Devor

Name: Tevi Devor

EXHIBIT A

42390.P8254

CLAIMS

1. A machine-implemented method comprising:

analyzing one or more instructions of a first program; and

modifying the first program to expand a register set for a routine in the first program transparently to execution of the first program that includes adding one or more registers to the register set, wherein the one or more registers of an expanded register set are used by a second program to store data used to analyze the execution of the first program.

2. The method of claim 1, comprising:

identifying one or more register moves for the expanded register set; and

modifying the first program to perform the identified one or more register moves.

3. The method of claim 2, wherein the identifying comprises:

defining one or more move chains for the expanded register set, and

identifying a sequence of one or more register moves based on the defined one or more move chains.

4. The method of claim 1, wherein the modifying the first program comprises modifying the first program to expand a register set for a callee routine of the first program.

5. The method of claim 4, comprising:

modifying the first program to expand a register set for a caller routine that is to call the callee routine.

6. The method of claim 5, wherein the modifying the first program to expand a register set for the callee routine comprises modifying the first program to expand a register set that includes one or more registers of the register set for the caller routine.

7. The method of claim 5, comprising:

identifying one or more register moves for the register set of the caller routine; and

modifying the first program to perform the identified one or more register moves prior to or upon returning from the callee routine to the caller routine.

8. The method of claim 5, comprising:

identifying a register move from a register added to the register set for the caller routine to a register added to the register set for the callee routine; and

modifying the first program to perform the identified register move.

10. A machine-readable medium having instructions that, if executed by a machine, cause the machine to perform a method comprising:

analyzing one or more instructions of a first program; and

modifying the first program to expand a register set for a routine in the first program transparently to execution of the first program that includes adding one or more registers to the register set, wherein the one or more registers of an expanded register set are used by a second program to store data used to analyze the execution of the first program.

11. The machine-readable medium of claim 10, wherein the method comprises:

identifying one or more register moves for the expanded register set; and

modifying the first program to perform the identified one or more register moves.

12. The machine-readable medium of claim 11, wherein the identifying comprises:

defining one or more move chains for the expanded register set, and

identifying a sequence of one or more register moves based on the defined one or more move chains.

13. The machine-readable medium of claim 10, wherein the modifying the first program comprises modifying the first program to expand a register set for a callee routine of the first program.

14. The machine-readable medium of claim 13, wherein the method comprises:

modifying the first program to expand a register set for a caller routine that is to call the callee routine.

15. The machine-readable medium of claim 14, wherein the modifying the first program to expand a register set for the callee routine comprises modifying the first program to expand a register set that includes one or more registers of the register set for the caller routine.

16. The machine-readable medium of claim 14, wherein the method comprises:

identifying one or more register moves for the register set of the caller routine; and

modifying the first program to perform the identified one or more register moves prior to or upon returning from the callee routine to the caller routine.

17. The machine-readable medium of claim 14, wherein the method comprises:

identifying a register move from a register added to the register set for the caller routine to a register added to the register set for the callee routine; and

modifying the first program to perform the identified register move.

19. A system comprising:

a processor to execute instructions; and

a medium having instructions to analyze one or more instructions of a first program and to modify the first program to expand a register set for a routine in the first program transparently to execution of the first program that includes adding one or more registers to the register set, wherein the one or more registers of an expanded register set are used by a second program to store data used to analyze the execution of the one or more instructions of the first program.

20. The system of claim 19, the medium having instructions to identify one or more register moves for the expanded register set and to modify the first program to perform the identified one or more register moves.

21. The system of claim 20, the medium having instructions to define one or more move chains for the expanded register set and to identify a sequence of one or more register moves based on the defined one or more move chains.

22. The system of claim 19, the medium having instructions to modify the first program to expand a register set for a callee routine of the first program.

23. The system of claim 22, the medium having instructions to modify the first program to expand a register set for a caller routine that is to call the callee routine.

24. The system of claim 23, the medium having instructions to modify the first program to expand a register set that includes one or more registers of the register set for the caller routine.

25. The system of claim 23, the medium having instructions to identify one or more register moves for the register set of the caller routine and to modify the first program to perform the identified one or more register moves prior to or upon returning from the callee routine to the caller routine.

26. The system of claim 23, the medium having instructions to identify a register move from a register added to the register set for the caller routine to a register added to the register set for the callee routine and to modify the first program to perform the identified register move.

EXHIBIT B

12620

REDACTED
INTEL INVENTION DISCLOSURE

INTEL CONFIDENTIAL

[redacted] P8254

DATE: [redacted] Architecture / MPG

It is important to provide accurate and detailed information on this form. The information will be used to evaluate your invention for possible filing as a patent application. When completed, please return this form to the **Legal Department at JF3-147**. If you have any questions, please call **264-0444 or 264-1476.**

1. Inventor: Baraz (Last Name) Leonid (First Name) (Middle Initial)
Phone +972-4-8655546 M/S: IDC-1D Fax # +972-4-8655938
Citizenship: ISRAEL WWID 10122409
Home Address: HaOren 81, Givat-Ram, City Kiryat-Ata State ISRAEL Zip
Group: (e.g. TMG, NBG, CEG) Btrans Division Name MPG Subdivision
Supervisor* Yaron Sheffer WWID 10022347 Phone +972-4-8655759 M/S: IDC-1D

Inventor: Devor (Last Name) Tevi (First Name) (Middle Initial)
Phone +972-4-8655675 M/S: IDC-1D Fax # +972-4-8655938
Citizenship: ISRAEL WWID 10022305
Home Address: Hashoftim 9a City Zichron Yaacov State ISRAEL Zip
Group: (e.g. TMG, NBG, CEG) Btrans Division Name MPG Subdivision
Supervisor* Yaron Sheffer WWID 10022347 Phone +972-4-8655759 M/S: IDC-1D

(PROVIDE SAME INFORMATION AS ABOVE FOR EACH ADDITIONAL INVENTOR)

2. Title of Invention: Method for Adding Integer Registers to an Existing Register Stack Frame in IA64

3. What technology/product/process (code name) does it relate to (be specific if you can):
Dynamic Optimization, Dynamic instrumentation

4. Stage of development (i.e. % complete, simulations done, test chips if any, etc.)
Implemented in prototype Dynamic Optimizer that is under development

5. (a) Has a description of your invention been, or will it shortly be, published outside Intel:
NO: NO YES: _____ If YES, was the manuscript submitted for pre-publication approval? _____
IDENTIFY THE PUBLICATION AND THE DATE PUBLISHED: _____

(b) Has your invention been used/sold or planned to be used/sold by Intel or others?
NO: NO YES: _____ DATE WAS OR WILL BE SOLD: _____

(c) Does this invention relate to technology that is or will be covered by a SIG (special interest group)/standard/ or specification?
NO: NO YES: _____ Name of SIG/Standard/Specification: _____

(d) If the invention is embodied in a semiconductor device, actual or anticipated date of tapeout? _____

(e) If the invention is software, actual or anticipated date of any beta tests outside Intel None

6. Was the invention conceived or constructed in collaboration with anyone other than an Intel blue badge employee or in performance of a project involving entities other than Intel, e.g. government, other companies, universities or consortia? NO: NO YES: _____ Name of individual or entity: _____

7. Is this invention related to any other invention disclosure that you have recently submitted? If so, please give the title and inventors: NO

INTEL CONFIDENTIAL

[redacted] **PLEASE READ AND FOLLOW THE DIRECTIONS ON HOW TO WRITE A DESCRIPTION OF YOUR INVENTION**

Please attach a page to this form, DATED AND SIGNED BY AT LEAST ONE PERSON WHO IS NOT A NAMED INVENTOR, to provide a description of the invention, and include the following information:

1. **Describe in detail what the components of the invention are and how the invention works.**

See below

2. **Describe advantage(s) of your invention over what is done now.**

Currently there is no existing applications we know of that do this

3. **YOU MUST include at least one figure illustrating the invention. If the invention relates to software, include a flowchart or pseudo-code representation of the algorithm.**

DONE. See below

4. **Value of your invention to Intel (how will it be used?).**

Will be used in Intel IA64 dynamic optimizer.

5. **Identify the closest or most pertinent prior art that you are aware of.**

NONE

6. **Who is likely to want to use this invention or infringe the patent if one is obtained and how would infringement be detected?**

Other applications that dynamically modify executables. Would be detected by code inspection

***HAVE YOUR SUPERVISOR READ, DATE AND SIGN COMPLETED FORM**

DATE: [redacted] SUPERVISOR: Yaron Sheffer_____

BY THIS SIGNING, I (SUPERVISOR) ACKNOWLEDGE THAT I HAVE READ AND UNDERSTAND THIS DISCLOSURE, AND RECOMMEND THAT THE HONORARIUM BE PAID

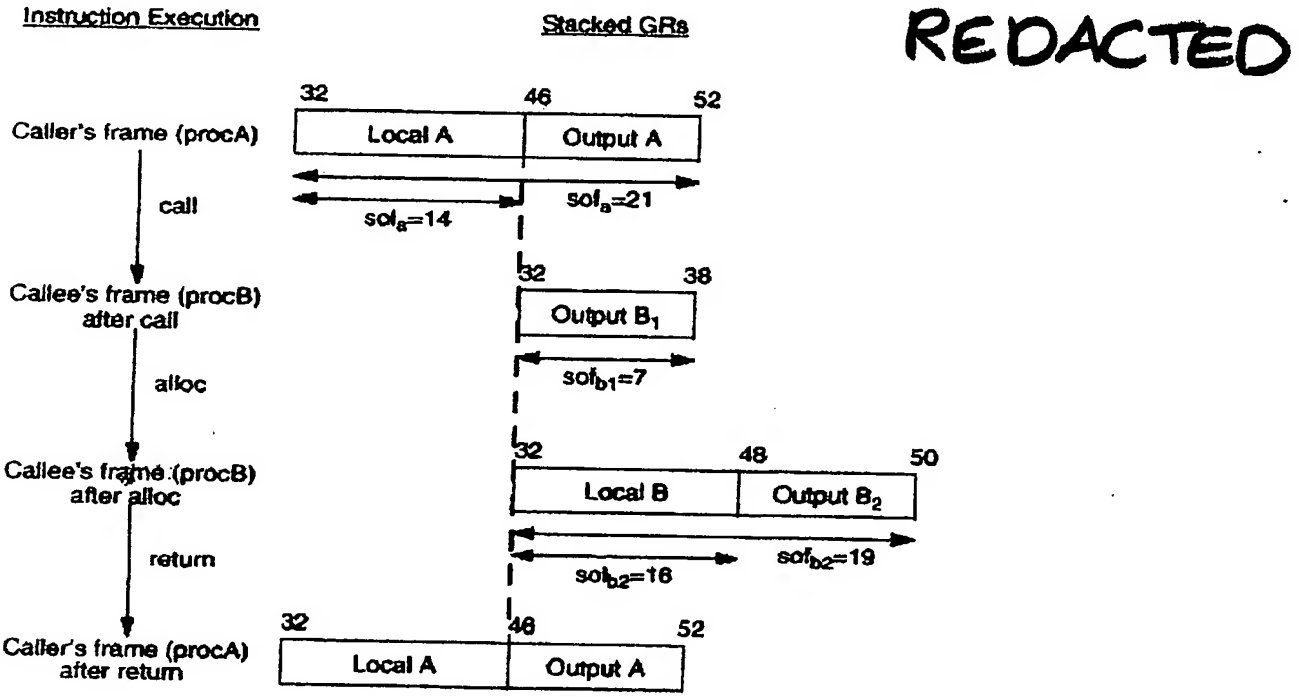
PATENT DECRIPTION: REDACTED

Applications that modify other applications (e.g. dynamic instrumenters, dynamic optimizers), may need to use registers for their own purposes in a manner that will be transparent to the source application being modified. This requires these applications to either save and restore registers in use by the source application – or to find registers that are free within a relevant scope in the source application. A method for obtaining registers in the IA64 architecture, by dynamically expanding the register frames of functions in the source application will be presented here.

The IA64 Architecture implements a per-function integer register stack. Each function may allocate itself a register stack frame of up to 96 integer stack registers beginning at GR32 through GR128. The frame is further partitioned into two variable-size areas: the local area and the output area. Immediately after a call-type branch, a new frame is created for the callee. The size of the local area of the newly activated frame is zero and the size of the output area is equal to the size of the caller's output area and overlays the caller's output area.

The callee can resize the frame using the IA64 **alloc** instruction which specifies immediate values that determine the size of the frame and the size of the locals. When a return-type branch is executed the frame reverts to the frame that was in effect at the caller before the call.

The following figure illustrates this when the function procA calls the function procB (note that *sof* is "start of frame" and *sol* is "start of locals"):



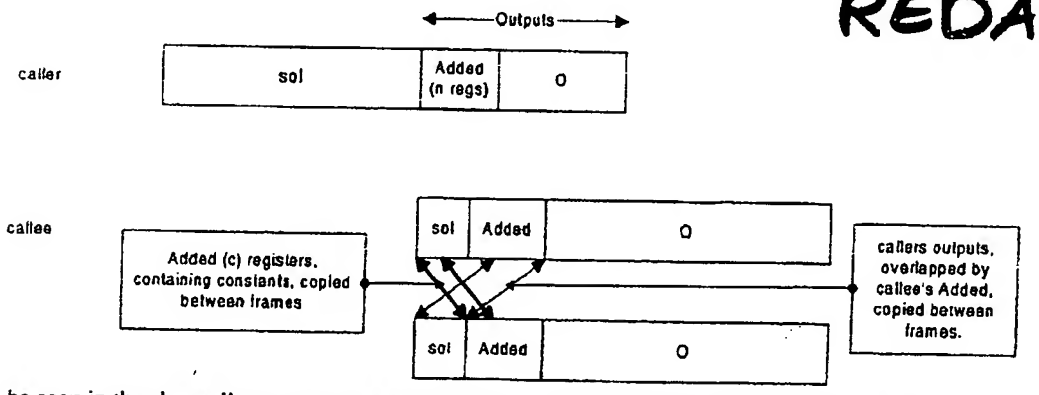
The full description of the register frame in IA64 is found at ftp://download.intel.com/design/IA64/Downloads/ADAG.pdf in section 4.1.1

As mentioned above, the modifying application will obtain registers for its own use by dynamically expanding the frame of the function in the source application. This is achieved by dynamically modifying the **alloc** instructions in the source application. The additional registers on the frame are transparent to the source application code, since these registers were not available when the source application code was generated. The output register area of the frame of each function is expanded by *n*, the number of additional registers desired. The first *n* registers in the output area of the frame are then available for use in the function. Furthermore, some of these registers may be used to hold values that are constant throughout the thread execution, and should maintain their values across calls, while others may not be required to do so. We denote by *c* the number of [illegible]ers used to hold constant values . c < n and these are the first *c* registers in the added registers. This is implemented as foll

1. References to an output register must be modified to access the corresponding output register after the expansion. This is done by adding n (where n is the number of registers that were added to the output area) to each output register reference.
2. Each function must observe the output registers of its caller in the same place as they would have been before the expansion, and the caller must be presented with those registers modified by the callee in the place it expects them. E.g. if the caller's output area is 8 registers starting at r50, and 5 registers are added to the output– The implementation of paragraph (1) will cause the caller to access r55 instead of r50. The callee, however will view the callee's r50 as the first register (r32) in its frame rather than r55 as it should. Hence the callee should copy the caller's r55 into its (the callee's) r32 immediately following the alloc. Prior to the return, the callee should copy its r32 back into the caller's r55.
3. In order to maintain those constant values that are held in the first c registers of the additional registers, these c registers are copied from the caller frame into the proper place in the callee frame at the beginning of the callee, and from the callee frame back to the caller frame prior to return from the callee. [this was mentioned before]

Copying in the direction of the call is done immediately after the alloc in the callee. Copying in the direction of the return is done immediately before the return. This is illustrated as follows:
The diagram should mention the parameters "n" and "c".

REDACTED



As can be seen in the above diagram some registers must be copied forward in the frame and others backwards in the frame – and there can be overlap in these. An efficient copying sequence is generated by determining copy chains and then executing the copies in each chain in reverse order.
Copy chains are formed and processed as follows:

First step is to determine given sol, n and c the registers that must be copied and their detination. An array, regCopyInfoArr, whose index is the source register and contents in the destination register is used to hold this information.

```
Now build all copy chains:
regCopyInfoArrayIndex = 0
while (1)
   /* find next element in regCopyInfoArray that is not on chain */
   while (regCopyInfoArrayIndex < maximum index in regCopyInfoArr &&
          (src register ,whose number is regCopyInfoArrayIndex, is not on a copy chain ||
           this src register does not have to be copied))
      regCopyInfoArrayIndex++;
   if (regCopyInfoArrayIndex exceeds max) /* all registers on copy chains */
      break out of while(1)
   BuildOneCopyChain starting from src register ,whose number is regCopyInfoArrayIndex

BuildOneCopyChain:
set src_register to the register specified by regCopyInfoArrayIndex
while (destination_register of src_register is not 0 and is not on chain)
   add destination_register of src_register, specified by
   regCopyInfoArr[regCopyInfoArrayIndex] to chain
   src_register = destination_register
if exited loop because destination_register was already on a chain and it was on this chain
   this is a loop chain
else
   the destination_register does not have to be copied and can be used in the sequence of copies    loop
```

copy chain

Now each chain is used to generate a sequence of copies:
the copies are executed in reverse order on the chain –
the second last register is moved to the last, the third last to the second last and so forth.
if the copy chain is a loop
the last register of one of the non loop copy chains is used to initiate the copy sequence

The destination of the IA 64 alloc instruction may be one of the local registers that is to be overwritten by caller's output – in this case the destination of the alloc instruction will be modified to be that output register that will be written to the local specified in the alloc instruction.

REDACTED

EXHIBIT C

AWARD MEMO

[redacted]

From: patent.database.group@intel.com
Sent: [redacted]
To: Devor, Tevi
Subject: AWARD MEMO

To: TEVI DEVOR
E-mail: tevi.devor@intel.com
Employee No.: 10022305

From: Leo Novakoski
Phone: (408) 765-5334

Subject: AWARD PAYMENTS

Thank you for your recent submission of the following invention disclosure:

METHOD FOR ADDING INTEGER REGISTERS TO AN EXISTING REGISTER STACK FRAME IN IA64
Disclosure #: 12620 Award, in the amount of $100*

IP Committee assigned: ARCHITECTURE

By copy of this letter, we are requesting that employee services provide you with award in the amount of $100 (tax protected)*

Your invention(s) will be reviewed at the next quarterly intellectual property committee meeting for your group to determine whether a patent application(s) will be filed. You will be notified of the outcome of this evaluation as soon thereafter as possible. If you have any questions regarding this award or Intel's patent process, please contact Janice Boulden at 503-264-0444.

*Please note that if you signed up to have all of your employee payables auto-deposited, then this award will be auto-deposited. Otherwise, this will be issued as a separate "live" check mailed to your home address.

PLEASE NOTE: Our "new" electronic inventor notifications save Intel substantial time and money but do not currently have the capability to copy your manager. Please feel free to forward to your manager.

For future update information, please visit our web site at law.intel.com/PPG2. You will need to provide your MAD log-in name and password.

mad#/username
password

REDACTED

[redacted]

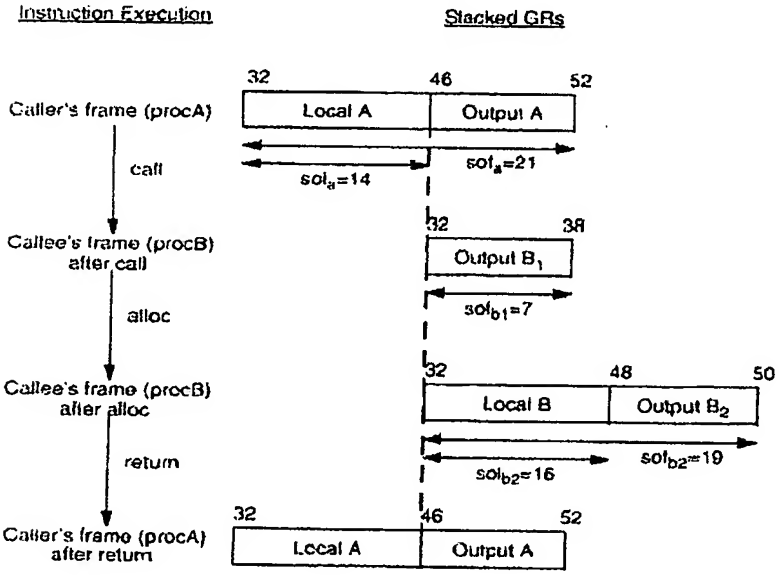
Adding Integer Registers to an Existing Register Stack Frame in IA64

Applications that modify other applications (e.g. dynamic instrumenters, dynamic optimizers), may need to use registers for their own purposes in a manner that will be transparent to the source application being modified. This requires these applications to either save and restore registers in use by the source application – or to find registers that are free within a relevant scope in the source application. A method for obtaining registers in the IA64 architecture, by dynamically expanding the register frames of functions in the source application will be presented here.

The IA64 Architecture implements a per-function integer register stack. Each function may allocate itself a register stack frame of up to 96 integer stack registers beginning at GR32 through GR128. The frame is further partitioned into two variable-size areas: the local area and the output area. Immediately after a call-type branch, a new frame is created for the callee. The size of the local area of the newly activated frame is zero and the size of the output area is equal to the size of the caller's output area and overlays the caller's output area.

The callee can resize the frame using the IA64 alloc instruction which specifies immediate values that determine the size of the frame and the size of the locals. When a return-type branch is executed the frame reverts to the frame that was in effect at the caller before the call.

The following figure illustrates this when the function procA calls the function procB (note that *sof* is "start of frame" and *sol* is "start of locals"):

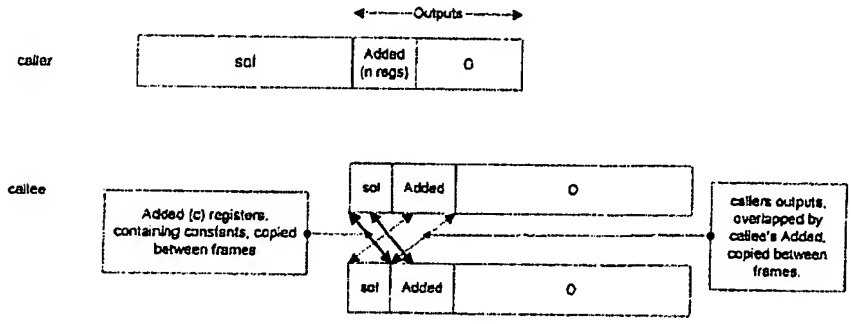


The full description of the register frame in IA64 is found at ftp://download.intel.com/design/IA64/Downloads/ADAG.pdf in section 4.1.1

As mentioned above, the modifying application will obtain registers for its own use by dynamically expanding the frame of the function in the source application. This is achieved by dynamically modifying the alloc instructions in the source application. The additional registers on the frame are transparent to the source application code, since these registers were not available when the source application code was generated. The output register area of the frame of each function is expanded by *n*, the number of additional registers desired. The first *n* registers in the output area of the frame are then available for use in the function. Furthermore, some of these registers may be used to hold values that are constant throughout the thread execution, and should maintain their values across calls, while others may not be required to do so. We denote by *c* the number of registers used to hold constant values . c < n and these are the first *c* registers in the added registers. This is implemented as follows:

1. References to an output register must be modified to access the corresponding output register after the expansion. This is done by adding n (where n is the number of registers that were added to the output area) to each output register reference.
2. Each function must observe the output registers of its caller in the same place as they would have been before the expansion, and the caller must be presented with those registers modified by the callee in the place it expects them. E.g. if the caller's output area is 8 registers starting at r50, and 5 registers are added to the output– The implementation of paragraph (1) will cause the caller to access r55 instead of r50. The callee, however will view the callee's r50 as the first register (r32) in its frame rather than r55 as it should. Hence the callee should copy the caller's r55 into its (the callee's) r32 immediately following the alloc. Prior to the return, the callee should copy its r32 back into the caller's r55.
3. In order to maintain those constant values that are held in the first c registers of the additional registers, these c registers are copied from the caller frame into the proper place in the callee frame at the beginning of the callee, and from the callee frame back to the caller frame prior to return from the callee. [this was mentioned before]

Copying in the direction of the call is done immediately after the alloc in the callee. Copying in the direction of the return is done immediately before the return. This is illustrated as follows:
The diagram should mention the parameters "n" and "c".



As can be seen in the above diagram some registers must be copied forward in the frame and others backwards in the frame – and there can be overlap in these. An efficient copying sequence is generated by determining copy chains and then executing the copies in each chain in reverse order.
Copy chains are formed and processed as follows:

First step is to determine given sol, n and c the registers that must be copied and their detination. An array, [redacted] whose index is the source register and contents in the destination register is used to hold this information.

Now build all copy chains:

[redacted]

The destination of the [redacted]on may be one of the local registers that is to be overwritten by caller's output – in this case the destination of the alloc instruction will be modified to be that output register that will be written to the local specified in the alloc instruction.

REDACTED

EXHIBIT D

NOTIFICATION OF PATENT APPLICATION FILINGS

[redacted]

From: patent.database.group@intel.com
Sent: [redacted]
To: Devor, Tevi
Subject: NOTIFICATION OF PATENT APPLICATION FILINGS

To: TEVI DEVOR
E-mail: tevi.devor@intel.com
Employee No.: 10022305

From: Leo Novakoski
Phone: (408) 765-5334

Subject: NOTIFICATION OF PATENT APPLICATION FILINGS

I am pleased to inform you that a determination has been made to file a U.S. patent application(s) covering your invention(s) as follows:

12620
METHOD FOR ADDING INTEGER REGISTERS TO AN EXISTING REGISTER STACK FRAME IN IA64

A patent attorney will be assigned to prepare the application(s) and will be contacting you for more details on your disclosure(s). Please cooperate with the attorney in answering questions and providing support for your invention(s). The attorney will use this information to prepare a draft patent application(s).

Once a draft of the application is prepared, you will be asked to review the draft to ensure that the most current version of the invention(s) is disclosed and suggest revisions prior to filing the application(s) with the U.S. Patent and Trademark Office. It is essential that you make your review of the application(s) a priority as patent rights can be lost for failure to timely file. Please do not take more than three weeks to review your application(s).

An honorarium will be paid to you once the patent application(s) is filed. In the meantime, if you have any questions, please call me.

PLEASE NOTE: Our "new" electronic inventor notifications save Intel substantial time and money but do not currently have the capability to copy your manager. Please feel free to forward to your manager.

For future update information, please visit our web site at law.intel.com/PPG2. You will need to provide your MAD log-in name and password.

mad#/username
password

REDACTED

[redacted]

EXHIBIT E

RE: Patent Application

[redacted]

From: Matthew Fagan [Matthew_Fagan@bstz.com]
Sent: [redacted]
To: Devor, Tevi
Subject: RE: Patent Application

REDACTED

One hour earlier would be fine. I'll look forward to your call.

"Devor, Tevi" <tevi.devor@intel.com> [redacted]

To: "Devor, Tevi" <tevi.devor@intel.com>, Matthew Fagan/Bstz
cc: "Baraz, Leonid" <leonid.baraz@intel.com>
Subject: RE: Patent Application

Hi.

I am really sorry - but I have to leave work on Monay at 17:00 our time (I just found out). Is it possible to do this one hour earlier on Monday, else on Thursday?

Thanks,
Tevi.

-----Original Message-----
From: Devor, Tevi
Sent: [redacted]
To: 'Matthew Fagan'
Subject: RE: Patent Application

Hi,

Leonid and I will call you.

Tevi

-----Original Message-----
From: Matthew Fagan [mailto:Matthew_Fagan@bstz.com]
Sent: [redacted]
To: Devor, Tevi
Cc: leonid.baraz@intel.com
Subject: RE: Patent Application

[redacted]

RE: Patent Application

Hi Tevi,

That time is fine. Should I call you? Is your phone number still 972-4-8655675? If you'd prefer to call me, my direct number is 1-512-306-7655. Let me know what's easiest for you.

Matt

REDACTED

"Devor, Tevi" <tevi.devor@intel.com>[redacted]

To: Matthew Fagan/Bstz
cc: "Baraz, Leonid" <leonid.baraz@intel.com>
Subject: RE: Patent Application

Hi.

Sorry for the long delay - Is [redacted] your time ok?

Tevi.

-----Original Message-----
From: Matthew Fagan [mailto:Matthew_Fagan@bstz.com]
Sent: [redacted]
To: tevi.devor@intel.com; leonid.baraz@intel.com
Subject: Patent Application

Reminder.

---------- Forwarded by Matthew Fagan/Bstz on [redacted] PM ----------

Matthew Fagan
[redacted]

To: tevi.devor@intel.com, leonid.baraz@intel.com
cc:
Subject: Patent Application

---------- Forwarded by Matthew Fagan/Bstz on [redacted] PM ----------

[redacted]

RE: Patent Application

Matthew Fagan
[redacted]

REDACTED

To: tevi.devor@intel.com, leonid.baraz@intel.com
cc:
Subject: Patent Application

Hi Tevi and Leonid,

I am the patent attorney assigned to prepare the patent application covering your invention entitled "Method for Adding Integer Registers to an Existing Register Stack Frame in IA64". Sorry about the delay in getting back to you.

Please let me know when you would be available to talk about your invention. I believe you are eight hours ahead of us. I am in the US Central Time zone which is two hours ahead of the US Pacific time zone. I would prefer anytime from 8 to 11 AM my time which is 4 to 7 PM your time.

I look forward to hearing from you.

Matt

****
Matthew C. Fagan
BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN, L.L.P.
1501 South Mo Pac Expressway
Suite 250
Austin, Texas 78746-7596

Telephone: (512) 330-0844
Facsimile: (512) 330-0476

E-Mail: mailto:matthew_fagan@bstz.com
Website: http://www.bstz.com
****

****
Matthew C. Fagan
BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN, L.L.P.
1501 South Mo Pac Expressway
Suite 250
Austin, Texas 78746-7596
[redacted]

Telephone: (512) 330-0844
Facsimile: (512) 330-0476

E-Mail: mailto:matthew_fagan@bstz.com
Website: http://www.bstz.com

****

REDACTED

****

Matthew C. Fagan
BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN, L.L.P.
1501 South Mo Pac Expressway
Suite 250
Austin, Texas 78746-7596

Telephone: (512) 330-0844
Facsimile: (512) 330-0476

E-Mail: mailto:matthew_fagan@bstz.com
Website: http://www.bstz.com

****

****

Matthew C. Fagan
BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN, L.L.P.
1501 South Mo Pac Expressway
Suite 250
Austin, Texas 78746-7596

Telephone: (512) 330-0844
Facsimile: (512) 330-0476

E-Mail: mailto:matthew_fagan@bstz.com

RE: Patent Application

Website: http://www.bstz.com

****

REDACTED

****

Matthew C. Fagan
BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN, L.L.P.
1501 South Mo Pac Expressway
Suite 250
Austin, Texas 78746-7596

Telephone: (512) 330-0844
Facsimile: (512) 330-0476

E-Mail: mailto:matthew_fagan@bstz.com
Website: http://www.bstz.com

****

This electronic message and its accompanying attachments (if any) contain information from the law firm of Blakely Sokoloff Taylor & Zafman LLP that is confidential and/or subject to attorney-client privilege. If you are not the intended recipient, be aware that any disclosure, copying, distribution, or use of the contents of this information is prohibited. If you have received this message in error, please notify the above attorney by telephone immediately.